Plans and Status of the Oxford L3 Si-tracking effort

Our work concept for the next 3 month

- Theme:
  - Try to improve/accelerate L3-Si tracking by Regionalisation
- Dogmas:
  - Keep it simple
  - Use as much existing offline code as possible
  - Be minimally invasive to existing code

Concept continued

- Strategy:
  - L3-Si-tracking regions at L3 are provided by L2-objects or "COT-approved" L2-objects (outside-in-tracking).
    - XFT-tracks = helix in r-ϕ-plane with errors, no limit in z
    - SVT-Tracks = as XFT, ϕ in one wedge, z in one barrel
    - μ-stubs = full (?) helix with errors (ask muon group)
    - em-clusters = $\eta\phi$-square at R_{cal} pointing to luminous region
    - jets? = ?Can a jet request regional tracking?
    - COT-track (confirming above) = full helix with errors
  - Caveat: Some triggers are foreseen to request additional global track reconstruction (multi-jet, E_t^{miss} + heavy-flavour)

Concept continued

- Strategy continued:
  - Best to regionalise as early as possible in reconstruction chain:

SIXD_Bank ⇒ SiStripSet ⇒ ClusterData ⇒ Hitlist ⇒ TrackSet

List of Regions, one chain for each region

- Try different algorithms:
  OI-Tracking, Trackmongering
- Compare:
  speed, efficiency, quality = "purity + χ^2 - ghosts"
- need hit-MC-particle associations !!!

  - Reasons:
    - max. gain in speed (avoid work as early as possible)
    - minimally invasive, only SIXD_bank and its iterators know
    - simple L3 control = loop over all reagions (L2-Objects)

Problem

We have no L2 simulation → We have no source of regions

Workaround

(to be done)

Fake a L2 b-bbar event by:

- look into MC-Event and only accept if >= 2 tracks (P_t>2GeV)
- find all tracks > 2GeV with COT
- create regions around COT-tracks
- Run Si-tracking in loop over all tracks / regions
- compare with non-regional results

Status

Region Classes

- Describe two groups of “things”:
  - Regions as given by L2 triggers *L2-Regions*
  - Detector parts such as
    - *readout regions* (VRB, VRB +HDI, VRB+HDI+Strip#)
    - *Detector parts* for PR (wafers, strips)

  which can lie-within / intersect-with tracking Regions
- Both groups are “geometrical regions in the CDF detector”
- Both now how to intersect with each other
- “Simple” region (interval) delegates intersect to complex region (SVT-track)

Region Base Class

`TrackingRegion`

- pure base class for all regions
- allows passing of base class references (don't need to know what type of regions to expect)
- demands from descendants to implement:
  - intersect(TrackingRegion)
  - answer to "what_type_are_you"
  - print(), read(), <<, >>
- drawback = virtual function calls (slower)

Specific Region Classes

Elementary

- **`TrackingPhiRegion`**
  just an interval in phi,
  used for a SVX-wedge
- **`TrackingZ_Region`**
  as above but in Z
- all other intervals (r, eta, theta)

Composite

- **`TrackingR_PhiZ_Region`**
  - contains 3 elementary regions
  - intersects in named order r < phi < z
- **`COT_TrackRegion`**
  - list of several TrackingR_PhiZ_Regions (one for each Si-layer intersect)
  - created from another track (COT, XFT, SVT)
- One for each L2 trigger object that is allowed to seed a region

Implementation stuff

- Intesections needed for StripSet creation:
  - *Readout-Regions* intersect *L2-Regions*
  - only needed for **`ReadSIXD_Bank()`** from TRY
  - *Readout-Regions* identified by VRB/HDI/Strip #
  - should be FAST:
    - create LUT's during first event
    - LUT's map
      - VRB with **`TrackingPhiZ_Region`**
      - VRB+HDI with **`TrackingR_PhiZ_Region`**
      - optionally VRB+HDI+Strip# with **`TrackingR_PhiZ_Region`** (caveat: What about stereos?)
    - LUT's grouped in **`SIXD_LUT`** class which is nested in **`SIXD_Bank`**
    - **`SIXD_LUT`** is static member of SIXD_Bank

More Implementation Stuff

- Invasions on existing code
  - **`TRY_Run2SiStripSet`** has defaulted region argument. default = **`TrackingWorld`** (intersect is always true with anything)
  - Region argument is passed on to the constructors of SIXD_Bank-Iterators (also have defaults) which use the LUTs and fill them on 1st event

Possible Extensions

- For OI-Tracking:
  - Start with a regionalised HitList
  - regionalise Pathfinder.getWaferIntersect() to avoid looking through all. Maybe best done by a limited wafer-set???
- Maybe add LUT's which work by DIGI code and Strip# for easier/faster search during any PR?